LPI exam 301 prep, Topic 304: Usage

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

Sean Walberg (sean@ertw.com)
Network engineer
Freelance

25 Mar 2008

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute Senior Level Linux Professional (LPIC-3) exam. In this fourth in a series of series of six tutorials, Sean walks you through searching your LDAP tree and using the command-line tools. You'll also learn how to set up Microsoft Outlook to query your LDAP tree.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The Linux Professional Institute (LPI) certifies Linux® system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also need to pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a

corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	Learn how to install, configure, and use the OpenLDAP software.
Topic 303	LPI exam 301 prep: Configuration	Learn how to configure the OpenLDAP software in detail.
Topic 304	LPI exam 301 prep: Usage	(This tutorial) Learn how to search the directory and use the OpenLDAP tools. See the detailed objectives.
Topic 305	LPI exam 301 prep: Integration and migration	Coming soon.
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), the following should be true:

- You should have several years of experience with installing and maintaining Linux on a number of computers for various purposes.
- You should have integration experience with diverse technologies and operating systems.
- You should have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role).
- You should know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- You should be able to use open source tools to measure capacity planning and troubleshoot resource problems.
- You should have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Active Directory.

- You should be able to plan, architect, design, build, and implement a full environment using Samba and LDAP as well as measure the capacity planning and security of the services.
- You should be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C).

To continue preparing for certification level 3, see the series developerWorks tutorials for LPI exam 301, as well as the entire set of developerWorks LPI tutorials.

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Configuration," the fourth of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you learn how to search the directory, how to use the command-line tools for searching and administration, and how to configure third-party applications to use your LDAP tree as a Whitepages service.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

Table 2 provides the detailed objectives for this tutorial.

Table 2. Configuration: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
304.1 Searching the directory	2	Use advanced options to search the LDAP directory
304.2 LDAP command-line tools	4	Use the various command-line tools to search, modify, and administer the LDAP server
304.3 Whitepages	1	Use your LDAP server as a Whitepages service for applications like Microsoft Outlook®

Prerequisites

To get the most benefit from this tutorial, you should have advanced knowledge of

Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the tutorials for the LPIC-1 and LPIC-2 exams.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in this tutorial, you'll need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Searching the directory

This section covers material for topic 304.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, learn how to:

- Use OpenLDAP search tools with basic options
- Use OpenLDAP search tools with advanced options
- Optimize LDAP search queries
- Use search filters and their syntax

The data in your tree is useful only if you can find entries when you need them. LDAP provides a powerful set of features that allow you to extract information from your tree.

The basics of search

To search a tree, you need four pieces of information:

1. Credentials on the server that holds the tree
2. A Distinguished Name (DN) on the tree to base your search on

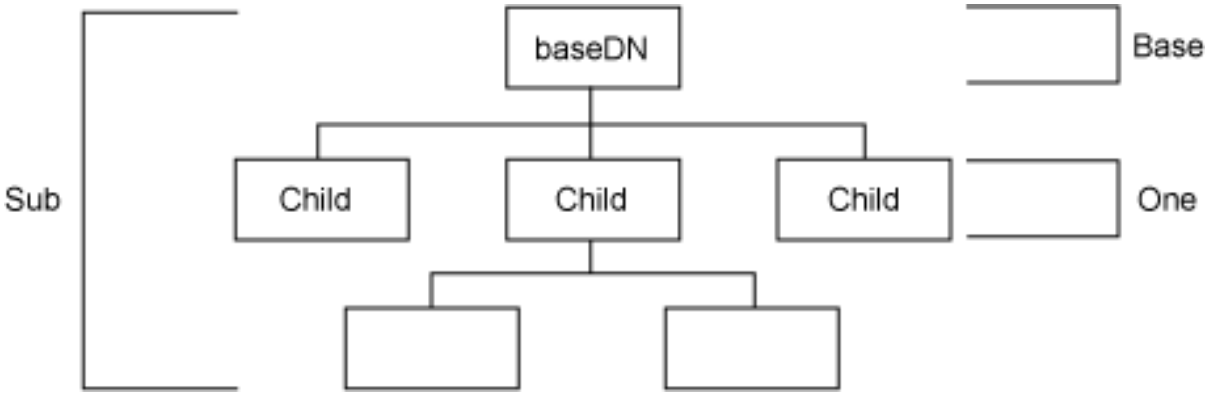
3. A search scope
4. A search filter

You have already learned about server credentials in previous tutorials from this series. The credentials can be nothing, which results in an anonymous bind, or they can be the DN of an entry along with a password. Implicit in this is that the server recognizes these credentials as valid, and is willing to allow you to search!

The DN that you base your search on is called the *base DN*. All results will be either the base DN itself or its children. If your base DN is `ou=people,dc=ertw,dc=com`, then you might find `cn=Sean Walberg,ou=people,dc=ertw,dc=com`, but you won't find `cn=Users,ou=Groups,dc=ertw,dc=com`, because it lies outside the base DN you were trying to search.

The scope determines which entries under the base DN will be searched. You may want to limit the scope because of performance reasons, or because only certain children of the base DN contain the information you want. The default search scope, *subordinate* (usually abbreviated as *sub*), includes the base DN and all children. You can search only the base DN with a *base* scope, such as when you want to test to see if an entry exists. The search scope called *one* searches only the base DN's immediate children and excludes any grandchildren and the base DN itself. Figure 1 shows a tree and the entries that would be included in the three different search scopes.

Figure 1. Three different search scopes



The most powerful (and complex) part of searching is the search filter. The credentials, base DN, and scope limit which entries are to be searched, but it is the *query* that examines each record and returns only the ones that match your criteria.

Simple search filters

Search filters are enclosed in parentheses. Within the parentheses is one

`attribute=value` pair. A simple search filter would be `(objectClass=inetOrgPerson)`, which will find any entries with an objectClass of `inetOrgPerson`. The attribute itself is not case sensitive, but the value may or may not be depending on how the attribute is defined in the schema. Recall from the first tutorial, LPI exam 301 prep: Concepts, architecture, and design, that the schema defines the attributes and their properties. One property of the attribute is if comparisons are case sensitive or not.

Substring searches are performed with the asterisk (*) operator. Search for `(Sean*)` to match anything beginning with `Sean`. The asterisk can go anywhere in the string, such as `(* Walberg)` to find anything ending in `Walberg`, or even `S*Wa*berg` to find anything starting with `S`, ending in `berg`, and having `Wa` somewhere in the middle. You might use this to find the author's name, not knowing if it is Sean or Shawn, or Walberg or Wahlberg.

The most generic form of the asterisk operator, `attribute=*` checks for the existence of the specified attribute. To find all the entries with defined e-mail addresses, you could use `(mail=*)`.

AND, OR, and NOT

You can perform logical AND and OR operations with the "&" and "|" operators respectively. LDAP search strings place the operator before the conditions, so you will see filters like those shown in Listing 1.

Listing 1. Sample search filters using AND and OR

```
(|(objectClass=inetOrgPerson)(objectClass=posixAccount))
(&(objectClass=*)(cn=Sean*)(ou=Engineering))
(&(|(objectClass=inetOrgPerson)(objectClass=posixAccount))(cn=Sean*))
```

The first search string in Listing 1 looks for anything with an objectClass of `inetOrgPerson` or `posixAccount`. Note that each component is still enclosed in parentheses, and that the OR operator (|) along with its two search options are also enclosed in another set of parentheses.

The second search string is similar, but starts with an AND operation instead of OR. Here, three different tests must be satisfied, and they all follow the ampersand in their own set of parentheses. The first clause, `objectClass=*`, matches anything with a defined objectClass (which should be everything, anyway). This search of all objectClasses is often used as a search filter when you want to match everything and are required to enter a filter. The second clause matches anything that starts with `Sean`.

The third search string shows both an AND and an OR used together in a filter that looks for anything with an objectClass of either `inetOrgPerson` or

, and a common name beginning with `Sean`.

The logical NOT is performed with the exclamation mark (!), much like the AND and OR. A logical NOT has only one argument so only one set of parentheses may follow the exclamation mark. Listing 2 shows some valid and invalid uses of NOT.

Listing 2. How to use, and how not to use, the logical NOT

```
(!cn=Sean)     # invalid, the ! applies to a filter inside ()
(!(cn=Sean))   # valid
(!(cn=Sean)(ou=Engineering))  # invalid, only one filter can be negated
(!(&cn=Sean*)(ou=Engineering))) # valid, negates the AND clause
```

In the fourth example of Listing 2, the negation is applied to an AND filter. Thus, that rule returns any entries that do not satisfy both of the AND clauses. Be careful when dealing with negation of composite filters, because the results are not always intuitive. The fourth example from Listing 2 will still return entries with an `ou` of `Engineering` if they don't have a common name starting with `Sean`. Both tests must pass for the record to be excluded.

Searching ranges

Often you need to search for a range of values. LDAP provides the <= and >= operators to query an attribute. Be careful that the equal sign (=) is included, because there are no < and > operators—you must also test for equality.

Not all integer attributes can be checked with the range operators. When in doubt, check the schema to make sure the attribute implements an ordering type through the `ORDERING` keyword. Recall from the first tutorial, LPI exam 301 prep: Concepts, architecture, and design, that an attribute is defined in the schema, and part of the definition includes how the server should sort the attribute.

Searching for close matches

An LDAP directory is often used to store names, which leads to spelling problems. "Sean" can also be "Shawn" or "Shaun." LDAP provides a "sounds-like" operator, "~=", which returns results that sound similar to the search string. For example, `(cn~=Shaun)` returns results that have a common name containing a word that sounds like "Shaun." Implicit in the sounds-like search is a substring search, such that `(cn=~Shaun)` will return results for `cn=Shawn Walberg`. The OpenLDAP implementation is not perfect, though; the same search will not return results for the "Sean" spelling.

Searching the DN

All the examples so far have focused on searching attributes, not searching on the distinguished name (DN) that identifies the record. Even though the leftmost

component of the DN, the relative DN (RDN), appears as an attribute and is therefore searchable, the search filters presented so far will not look at the rest of the DN.

Searching the DN is done through a specific query filter requiring an exact match. The format is `attribute:dn:=value`, where the attribute is the component of the DN you want to search, and the value is the search string (no wildcards allowed). For example, `(ou:dn:=People)` would return all the entries that have `ou=People` in the DN, including the container object itself.

Altering the matchingRule

By default, most strings, such as the common name, are case insensitive. If you want to override the matching rule, you can use a form similar to the DN search. A search such as `(ou:caseexactmatch:=people)` will match an organizational unit of "people", but not "People". Some common matching rules are:

- `caseIgnoreMatch` matches a string without regard for capitalization. Also ignores leading and trailing whitespace when matching.
- `caseExactMatch` is a string match that also requires similar capitalization between the two strings being searched.
- `octetStringMatch` is like a string match, but does not remove whitespaces, and therefore requires an exact, byte-for-byte, match.
- `telephoneNumberMatch` searches a telephone number, which has its own data type in LDAP.

You can also change the matching rule of a DN search by combining the DN search with the matching rule search. For example, `(ou:dn:caseexactmatch:=people)` searches for a DN containing the exact string "people".

These two types of searches, DN searches and matching rule searches, are also called *extensible searches*. They both require exact strings and do not allow wildcards.

Using ldapsearch

The command-line tool to search the tree is `ldapsearch`. This tool lets you bind to the directory in a variety of ways, execute one or more searches, and retrieve the data in LDIF format.

The default behavior of `ldapsearch` is:

- Attempt a Simple Authentication and Security Layer (SASL) authentication to the server
- Connect to the server at ldap://localhost:389
- Use `(objectClass=*)` as a search filter
- Read the search base from /etc/openldap/ldap.conf
- Perform a sub search; that is, include the search base and all children
- Return all user attributes, ignoring operational (internal usage) attributes
- Use extended LDAP Data Interchange Format (LDIF) for output
- Do not sort the output

Authenticating to the server

If you are not using SASL, then you need simple authentication using the `-x` parameter. By itself, `-x` performs an *anonymous bind*, which is a bind without any bind DN or password. Given the other defaults, `ldapsearch -x` will dump your entire tree, starting at the search base specified in /etc/openldap/ldap.conf. Listing 3 shows the usage of a simple anonymous search.

Listing 3. A simple anonymous search

```
$ ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# people, ertw.com
dn: ou=people,dc=ertw,dc=com
ou: people
description: All people in organization
objectClass: organizationalUnit
... output truncated ...
```

Listing 3 shows the header and first entry returned from a simple anonymous search. The first seven lines form the header, and in LDIF fashion, are commented starting with the hash mark (#). The first three lines identify the rest of the text as being extended LDIF and retrieved using LDAP version 3. The next line indicates that no base DN was specified and that a subtree search was used. The last two lines of text give the search filter which is everything and that all attributes were requested.

You may use the `-LLL` option to remove all the comments from your output.

After the header comes each entry; each entry starts with a header describing the entry and then the list of attributes, starting with the DN. Attributes are not sorted.

If you need to use a username and password to log in, use the `-D` and `-w` options to specify a bind DN and a password, respectively. For example, `ldapsearch -x D cn=root,dc=ertw,dc=com -w mypassword` will perform a simple authentication with the root DN username and password. You may also choose to type the password into a prompt that does not echo to the screen by using `-W` instead of `-w password`.

You may also connect to a different server by passing a Uniform Resource Identifier (URI) to the remote LDAP server using the `-H` option, such as `ldapsearch -x -H ldap://192.168.1.1/` to connect to an LDAP server at `192.168.1.1`.

Performing searches

Append your search filter to your command line in order to perform a search. You will likely have to enclose the filter in quotes to protect special characters in the search string from being interpreted by the shell. Listing 4 shows a simple search on the common name.

Listing 4. A simple search from the command line

```
$ ldapsearch -LLL -x '(cn=Fred Smith)'
dn: cn=Fred Smith,ou=people,dc=ertw,dc=com
objectClass: inetOrgPerson
sn: Smith
cn: Fred Smith
mail: fred@example.com
```

The search in Listing 4 uses the `-LLL` option to remove comments in the output and the `-x` option to force simple authentication. The final parameter is a search string that looks for Fred Smith's entry. Note that the parentheses are used around the search, and that single quotes are used both to protect the parentheses as being interpreted as a subshell invocation, and because the search string contains a space which would cause the "Smith" to be interpreted as a separate argument.

Listing 4 returned all of Fred Smith's attributes. It is a waste of both client and server resources to retrieve all values of a record if only one or two attributes are needed. Add the attributes you want to see to the end of the `ldapsearch` command line to only request those attributes. Listing 5 shows how the previous search looks if you only wanted Fred's e-mail address.

Listing 5. Requesting Fred Smith's e-mail address

```
$ ldapsearch -LLL -x '(cn=Fred Smith)' mail
dn: cn=Fred Smith,ou=people,dc=ertw,dc=com
```

```
mail: fred@example.com
```

The `mail` attribute is appended to the command line from Listing 4, and the result is the distinguished name of the record found, along with the requested attributes.

`ldapsearch` looks to /etc/openldap/ldap.conf for a line starting with BASE to determine the search base, and failing that, relies on the server's `defaultsearchbase` setting. The search base is the point on the tree where searches start from. Only entries that are children of the search base (and the search base itself) will be searched. Use the `-b` parameter to specify a different search base, such as `ldapsearch -x -b ou=groups,dc=ertw,dc=com` to search the groups container from the ertw.com tree.

Altering how data is returned

LDAP can store binary data such as pictures. The `jpegPhoto` attribute is the standard way to store a picture in the tree. If you retrieve the value of the attribute from the command line, you will find it is base64 encoded. The `-t` parameter is used to save any binary attributes into a temporary file. Listing 6 shows how to use this parameter.

Listing 6. Saving binary attributes on the file system

```
$ ldapsearch  -LLL  -x 'cn=joe*' jpegphoto | head
dn: cn=Joe Blow,ou=people,dc=ertw,dc=com
jpegPhoto::
/9j/4AAQSkZJRgABAQEASABIAAD//gAXQ3JlYXRlZCB3aXRoIFRoZSBHSU1Q/9sAQw
... output continues for 1300+ lines ...

$ ldapsearch  -LLL -t -x '(cn=joe*)' jpegphoto
dn: cn=Joe Blow,ou=people,dc=ertw,dc=com
jpegPhoto:< file:///tmp/ldapsearch-jpegPhoto-VaIjkE

$ file /tmp/ldapsearch-jpegPhoto-VaIjkE
/tmp/ldapsearch-jpegPhoto-VaIjkE: JPEG image data, JFIF standard 1.01,
comment: \
  "Created with The GIMP\377"
```

Listing 6 shows two searches for anyone with a name beginning with "Joe," and only retrieving the `jpegPhoto` attribute. The first try does not use the `-t` parameter, and therefore the value of `jpegPhoto` is shown on the console in base64 format. The usefulness of this is limited at the command line, so the second try specifies `-t` on the command line. This time the value of `jpegPhoto` is a URI to a file (you may change the directory with the `-T` option). Finally, the returned file is inspected, and indeed, it is the binary version of the picture that can be viewed.

By default, `ldapsearch` prints out results in the order they were received in from the server. You can sort the output with the `-S` parameter, passing it the name of the attribute you want to sort on. For sorting on multiple attributes, separate the attributes with a comma (,).

Section 3. LDAP command-line tools

This section covers material for topic 304.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 4.

In this section, learn how to:

- Use the ldap* tools to access and modify the directory
- Use the slap* tools to access and modify the directory

Several tools are provided with OpenLDAP to manipulate the directory and administer the server. You are already familiar with `ldapsearch`, which was covered in the previous section. The commands beginning with `ldap` are for users of the tree, the commands beginning with `slap` are for administrators.

Tree manipulation tools

The commands in this section are for manipulating the tree, either by changing data or reading data. `ldapsearch` falls into this category, too. To use these commands, you need to authenticate to the server.

ldapadd and ldapmodify

These two commands are used to add and change entries in the tree. You may recall from the first tutorial, LPI exam 301 prep: Concepts, architecture, and design in the series that the LDAP Data Interchange Format (LDIF) can be used to add, change, and delete data from the tree. Listing 7 shows a sample of some LDIF to add an entry.

Listing 7. LDIF to add an entry to the tree

```
dn: cn=Sean Walberg,ou=people,dc=ertw,dc=com
objectclass: inetOrgPerson
cn: Sean Walberg
cn: Sean A. Walberg
sn: Walberg
homephone: 555-111-2222
```

Listing 7 begins with a description of the distinguished name of the entry. This entry will end up under the `ou=people,dc=ertw,dc=com` container, and have a relative distinguished name of `cn=Sean Walberg`, which is obtained by splitting the

distinguished name (DN) after the first attribute/value pair. The entry has an `objectclass` of `inetOrgPerson`, which is a fairly generic type for any person belonging to an organization. Two variants of the common name follow, then a surname, and finally, a home phone number.

Implicit in Listing 7 is that this is an addition to the tree, as opposed to a change or deletion. Recall that LDIF files can specify the `changetype` keyword, which tells the reader what to do with the data.

The `ldapadd` command is used to process this LDIF file. If Listing 7 were stored as "sean.ldif", then `ldapadd -x -D cn=root,dc=ertw,dc=com -w mypass -f sean.ldif` would be one way to add the new entry to the tree. The `-x -D cn=root,dc=ertw,dc=com -w mypass` part of the command should be familiar from the earlier discussion of `ldapsearch`, as a way to authenticate to the tree with simple authentication and the all-powerful root DN. All the `ldap` commands in this section use the same parameters to authenticate to the tree, so you will see this form repeated.

`ldapadd` is implemented as a symbolic link to `ldapmodify`, and when called as `ldapadd` is interpreted as `ldapmodify -a`. The `-a` parameter tells `ldapmodify` to assume a default `changetype` of `add`, which is used to add new entries to the tree. When called as `ldapmodify`, the assumption is that the default `changetype` is modify operation.

`ldapadd` (and `ldapmodify`) is an efficient way of loading bulk data into a server without shutting it down. LDIF files can contain many operations, and often it is easier to generate LDIF from whatever other data source you are trying to import than to write custom code to parse the data source and add it directly through LDAP.

ldapdelete

`ldapdelete`, like the name implies, deletes an entry from the tree. All entries are uniquely identified in the tree by their DN; therefore, `ldapdelete` deletes entries by DN, and not by any other query.

Besides the authentication parameters already discussed, `ldapdelete` can take its list of DNs to delete either from the command line or from a file. To delete from the command line, simply append the DNs to your command line, such as `ldapdelete -x -D cn=root,dc=ertw,dc=com -w mypass "cn=Sean Walberg,ou=people,dc=ertw,dc=com"`. If you have many entries to delete, you can place the DNs, one per line, in a file, and point `ldapdelete` to that file with `-f filename`.

Note that you can also delete entries through LDIF and the `ldapadd/ldapmodify` commands. The `ldapdelete` command is more convenient in many cases, but is not the only way of deleting entries.

ldapmodrdn

The `ldapmodrdn` command changes the relative distinguished name of the object, that is, the first attribute/value pair in the DN. This effectively renames the entry within the current branch of the tree. Unlike the LDIF `moddn changetype`, this command can only rename the entry, and cannot move it to another spot on the tree.

Usage of this command is simple: give it the authentication credentials, the DN of the entry, and the new RDN. Listing 8 shows an account being renamed from "Joe Blow" to "Joseph Blow".

Listing 8. Renaming an entry

```
$ ldapmodrdn -x -D cn=root,dc=ertw,dc=com -w dirtysecret \
        'cn=Joe Blow,ou=people,dc=ertw,dc=com' 'cn=Joseph Blow'
$ ldapsearch -LLL -x '(cn=Joseph Blow)'
dn: cn=Joseph Blow,ou=people,dc=ertw,dc=com
objectClass: inetOrgPerson
sn: Blow
cn: Joe Blow
cn: Joseph Blow
```

Note that the old RDN still appears as an attribute, that is, `cn: Joe Blow`. If you want the old RDN to be removed, add `-r` to your command line. This is the same as adding `deleteoldrdn: 1` to your LDIF code (which, curiously, is the default behavior for LDIF but not `ldapmodrdn`).

ldapcompare

`ldapcompare` allows you to compare a predetermined value to the value stored somewhere in the LDAP tree. An example will show how this works.

Listing 9. Using ldapcompare

```
$ ldapcompare -x "cn=Sean Walberg,ou=people,dc=ertw,dc=com" userPassword:mypassword
TRUE
$ ldapcompare -x "cn=Sean Walberg,ou=people,dc=ertw,dc=com" userPassword:badpassword
FALSE
```

In Listing 9, the `ldapcompare` command is run. After the authentication parameters are taken care of, the final two parameters are the DN to check and the attribute and value to check against. The DN in both the examples above is the listing for "cn=Sean Walberg". The attributes being checked in both cases are the `userPassword` attribute. When the proper password is given, `ldapcompare` prints the string TRUE and an error code of 6. If the value given doesn't match what's in the entry, then FALSE is sent to the console, and an error code of 5 is returned. The `-z` option prevents anything from being printed; the caller is expected to use the error

code to determine if the check was successful or not.

Even though the example in Listing 9 checked a password, any attribute can be used, including `objectClass`. If the attribute has multiple values, such as multiple common names or `objectClasses`, then the comparison is successful if one of them matches.

ldapwhoami

`ldapwhoami` allows you to test authentication to the LDAP server and to determine which DN you are authenticated against on the server. Simply call `ldapwhoami` with the normal authentication parameters, as shown in Listing 10.

Listing 10. A demonstration of ldapwhoami

```
$ ldapwhoami -x
anonymous
Result: Success (0)
$ ldapwhoami -x -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" -w
mypassword
dn:cn=Sean Walberg,ou=people,dc=ertw,dc=com
Result: Success (0)
$ ldapwhoami -x -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" -w badpass
ldap_bind: Invalid credentials (49)
```

The first example in Listing 10 shows a bind with no username or password. `Ldapwhoami` returns the string `anonymous` to indicate an anonymous bind, and also a status line indicating that the authentication was successful. The second example binds as a user's DN. This time the DN returned is the same one that was authenticated with. Finally, a bind attempt is made with invalid credentials. The result is an explanation of the problem.

`Ldapwhoami` is helpful for troubleshooting the configuration of the server, and also for manually verifying passwords. Access lists might get in the way of an `ldapsearch`, so using `ldapwhoami` instead can help you determine if the problem is credentials or access lists.

Administration tools

The commands beginning with `slap` are for administrators, and operate directly on the database files rather than through the LDAP protocol. As such, you will generally need to be root to use these commands, and in some cases, the server must also be shut down.

slapacl

`Slapacl` is a utility that lets the administrator test access lists against various

combinations of bind DN, entry, and attribute. For instance, you would use `slapacl` to test to see what access a particular user has on another user's attributes. This command must be run as root because it is reading the database and configuration files directly rather than using LDAP.

The usage of `slapacl` is best described through an example. In Listing 11, the administrator is testing to see what access a user has on his own password before implementing an ACL, and then again after implementing an ACL that is supposed to limit the access to something more secure.

Listing 11. Using slapacl to determine the effect of an ACL change

```
# slapacl -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" \
        -b "cn=Sean Walberg,ou=People,dc=ertw,dc=com" userPassword
authcDN: "cn=sean walberg,ou=people,dc=ertw,dc=com"
userPassword: read(=rscxd)

... change slapd.conf ...

# slapacl -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" \
        -b "cn=Sean Walberg,ou=People,dc=ertw,dc=com" userPassword
authcDN: "cn=sean walberg,ou=people,dc=ertw,dc=com"
userPassword: =wx

# slapacl -D "cn=Joseph Blow,ou=people,dc=ertw,dc=com" \
        -b "cn=Sean Walberg,ou=People,dc=ertw,dc=com" userPassword
authcDN: "cn=joseph blow,ou=people,dc=ertw,dc=com"
userPassword: =0
```

Two pieces of information are mandatory for the `slapacl` command. The first is the bind DN, which is the DN of the user you are testing access for. The second piece is the DN of the entry you are testing against. The bind DN is specified with `-D`, and the target DN is set with `-b`. You can optionally limit the test to a single attribute by including it at the end (like the `userPassword` example in Listing 11). If you don't specify an attribute, you will receive results for each attribute in the entry.

In the first command from Listing 11, the administrator is testing the `cn=Sean Walberg` entry to see what access he has against his own password. The result is read access. Recall from the third tutorial in this series, LPI exam 301 prep: Configuration, that users should be able to write and authenticate against their `userPassword` attribute, but not read it. After changing the ACLs, the test is performed again, and only the write and authenticate permissions are available. Finally, a test is performed to see what access Joseph Blow has on Sean Walberg's password; the result is that he has no access.

`Slapacl`is an effective way to test the results of ACL changes, and to debug ACL problems. It is particularly effective because it reads directly from the database and slapd.conf, so any changes made to slapd.conf are reflected in the output of `slapacl` and don't require a restart of `slapd`.

slapcat

`Slapcat` dumps the contents of the LDAP tree as LDIF to the standard output, or to a file if you use `-l filename`. You can optionally use the `-s` option to provide the starting DN, or `-a` to pass a query filter.

`Slapcat` operates directly on the database, and can be run while the server is still running. Only the bdb database types are supported.

slapadd

`Slapadd` is a bulk import tool that operates directly on the backend databases, which means slapd must be stopped to use this tool. It is designed to be used with the output from `slapcat`. `Slapadd` doesn't perform much validation on the input data, so it is possible to end up with branches of the tree that are separated. This would happen if some container objects weren't imported.

The input to `slapadd` is an LDIF file, such as the one generated by `slapcat`. The slapadd(8C) manpage suggests using `ldapadd` instead because of the data validation provided by the online variant. The manpage also notes that the output of `slapcat` is not guaranteed to be ordered in a way that is compatible with `ldapadd` (the container objects may come after the children in the output, and hence would fail validation). Using any filters in `slapcat` may also cause important data to be missing. Therefore, you should use slapadd only with LDIF produced by `slapcat`, and use `ldapadd` for any other LDIF.

After shutting down your LDAP server, you can just run the `slapadd` command and pipe your LDAP output to the standard input. If you want to read from a file, use the `-l` option. As with `slapcat`, only the bdb database types are supported.

slappasswd

`Slappasswd` is used to generate hashed passwords to be stored in the directory, or in slapd.conf. A common use is to use a hashed password for the rootdn's account in slapd.conf so that anyone looking at the configuration file can not determine the password. `Slappasswd` will prompt you for a password to hash if you don't provide any parameters, as shown in Listing 12.

Listing 12. Using slappasswd to hash a password

```
$ slappasswd
New password:
Re-enter new password:
{SSHA}G8Ly2+t/HMHJ3OWWE7LN+GRmZJAweXoE
```

You may then copy the entire string to the `rootpw` line in slapd.conf. `Slapd` recognizes the format of the password and understands that `{SSHA}` means that what follows is a SHA1 hash. Anyone who reads slapd.conf will not learn the root password.

The hashes generated by `slappasswd` can also be used in LDIF files used with `ldapadd` and `ldapmodify`, which will allow you to store secure one-way hashes of your password instead of a less secure plaintext or base64-encoded version.

slapindex

You may recall `slapindex` from the third tutorial in this series, LPI exam 301 prep: Configuration. After creating or changing an index with the `index` keyword in slapd.conf, you must rebuild your indexes, or `slapd` will return incorrect results. To rebuild your indexes, stop `slapd` and run `slapindex`. This may take a while depending on how many entries are in your databases, or as the manpage puts it, "This command provides ample opportunity for the user to obtain and drink their favorite beverage."

slaptest

`Slapdtest` simply checks to see if your slapd.conf file is correct. This is helpful because if you were to restart `slapd` with a bad configuration file, it would fail to start up until you fixed the file. `Slaptest` lets you perform a sanity check on your configuration file before restarting.

Using `slaptest` is as simple as typing `slaptest`. If the slapd.conf is correct, you will see `config file testing succeeded`. Otherwise, you will receive an error explaining the problem.

`Slaptest` also checks for the existence of various files and directories necessary for operation. During testing however, the author was able to find some configuration file errors that passed `slaptest`, but would still cause `slapd` to fail.

Section 4. Whitepages

This section covers material for topic 304.3 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to do:

- Plan whitepages services
- Configure whitepages services
- Configure clients to retrieve data from whitepages services

A *whitepages service* allows e-mail clients to retrieve contact information from an

LDAP database. By staying with common attribute names, such as those provided by the `inetOrgPerson objectClass`, you can get the most compatibility with e-mail clients. For example, both Microsoft Outlook and Evolution use the `mail` attribute to store the user's e-mail address, and the `givenName`, `displayName`, `cn`, and `sn` attributes to store various forms of the name.

Configuring e-mail clients for an LDAP directory

In theory, any mail client that supports LDAP can use your tree. You will need the following information configured in the client:

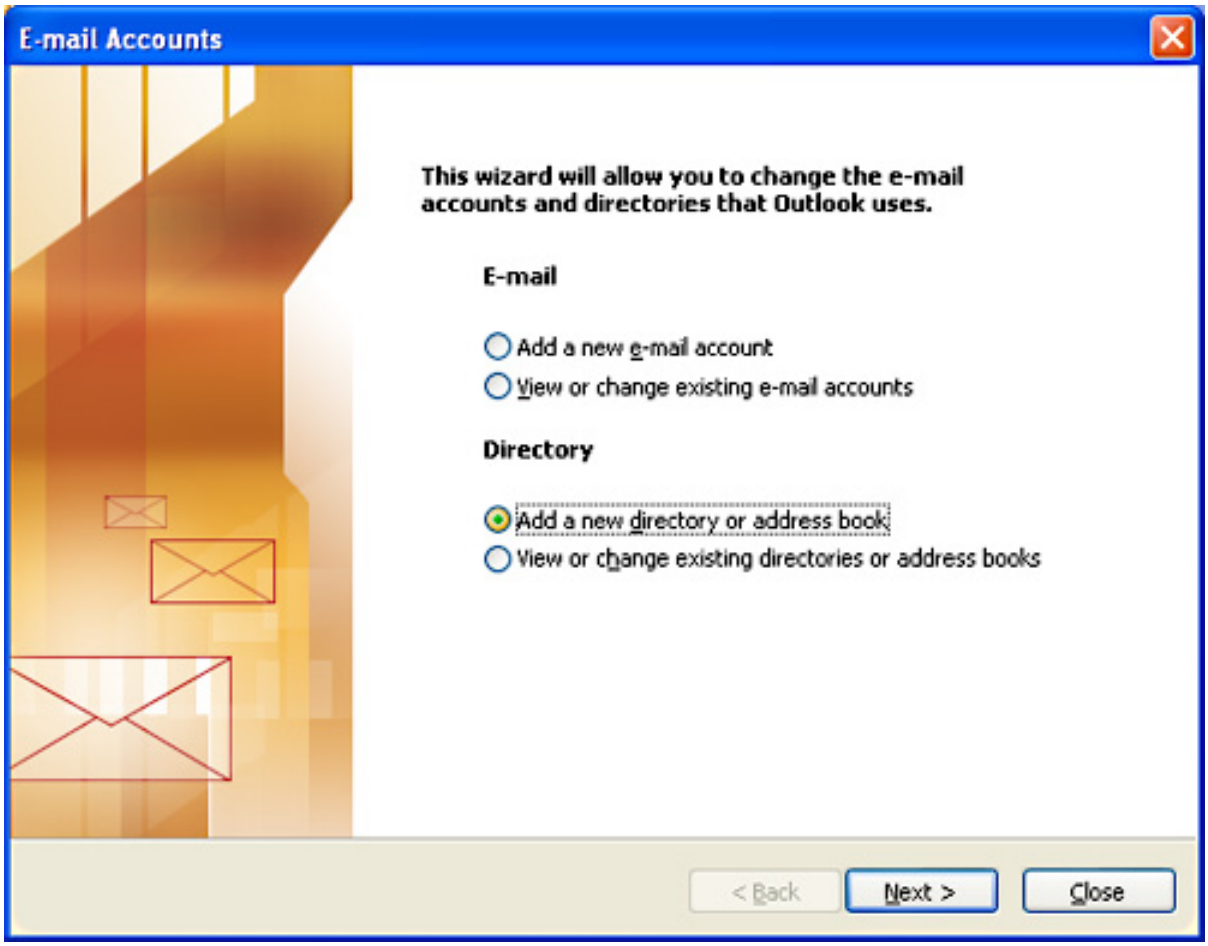
- The LDAP server's address or hostname
- Credentials to bind with, unless you are binding anonymously
- The base DN to search from
- A search filter, such as (mail=*), to weed out accounts without an e-mail address (optional)

Once you input the above information into your e-mail client, you should be able to search for contacts.

Configuring Microsoft Outlook for an LDAP directory

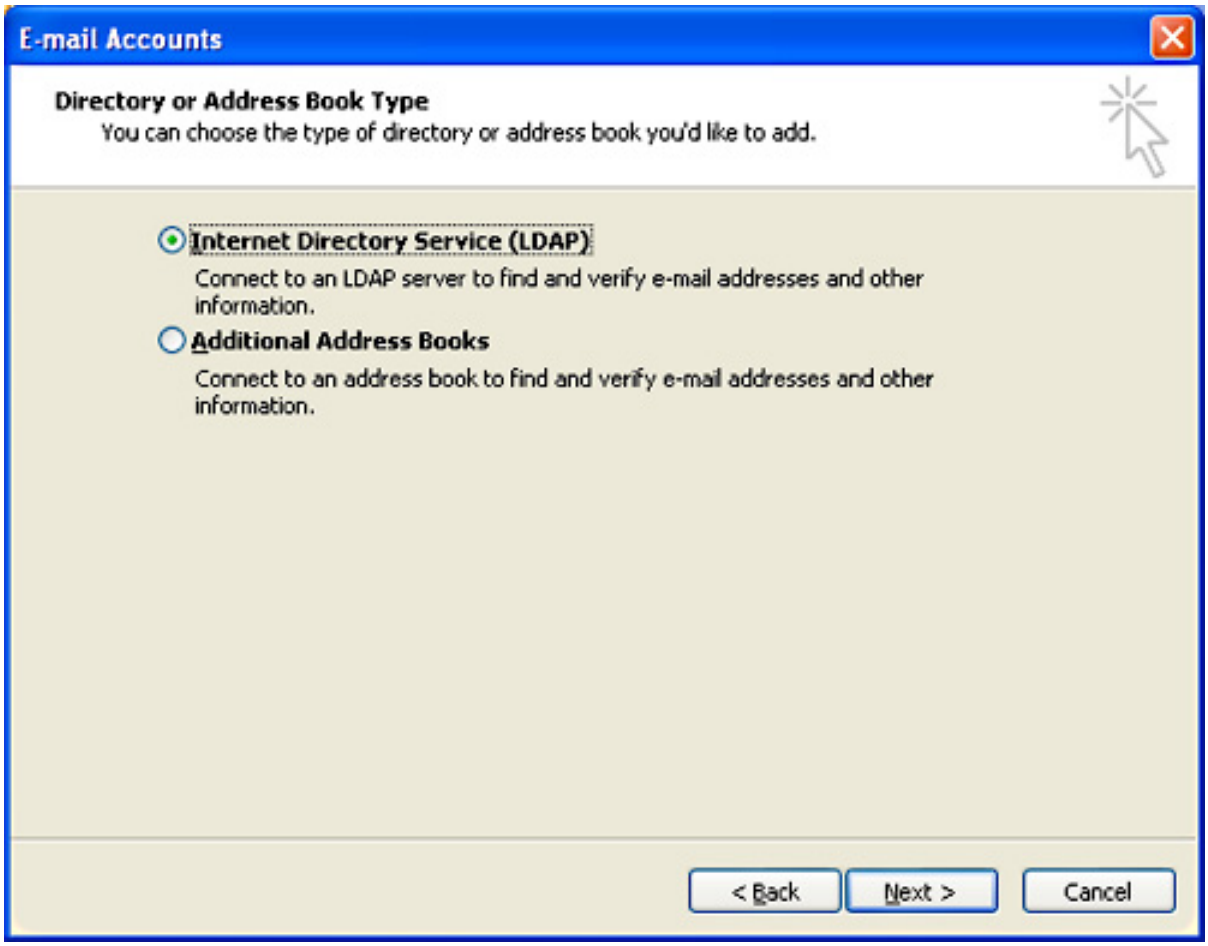
To configure Microsoft Outlook (tested on Outlook 2003), select **Tools > Email Accounts**. You will see a dialog similar to Figure 2.

Figure 2. Selecting the type of account to add



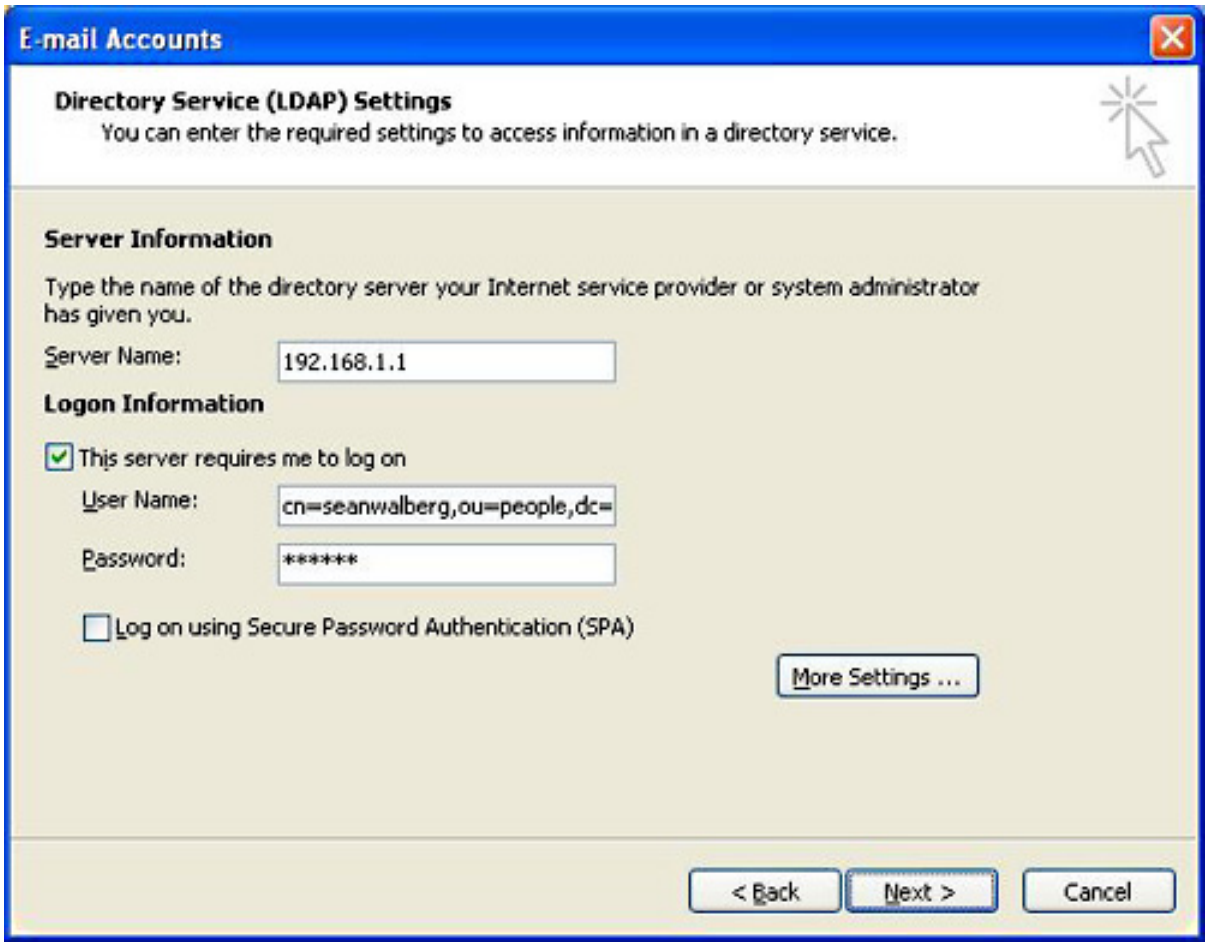
Select the option to add a new directory, and click **Next**. You will then see the dialog in Figure 3.

Figure 3. Selecting the type of directory to add



Select the option to add a new LDAP directory, and click **Next**. You will then see the dialog in Figure 4.

Figure 4. Specifying the LDAP server details



Enter the relevant details about your LDAP server in the dialog shown in Figure 4. The example shown uses user credentials to bind to the tree. You can use anonymous access if your server's configuration supports it.

After entering in the basic details, click **More Settings**, and you will be prompted for more information, as shown in Figure 5.

Figure 5. Adding advanced options to the LDAP server configuration

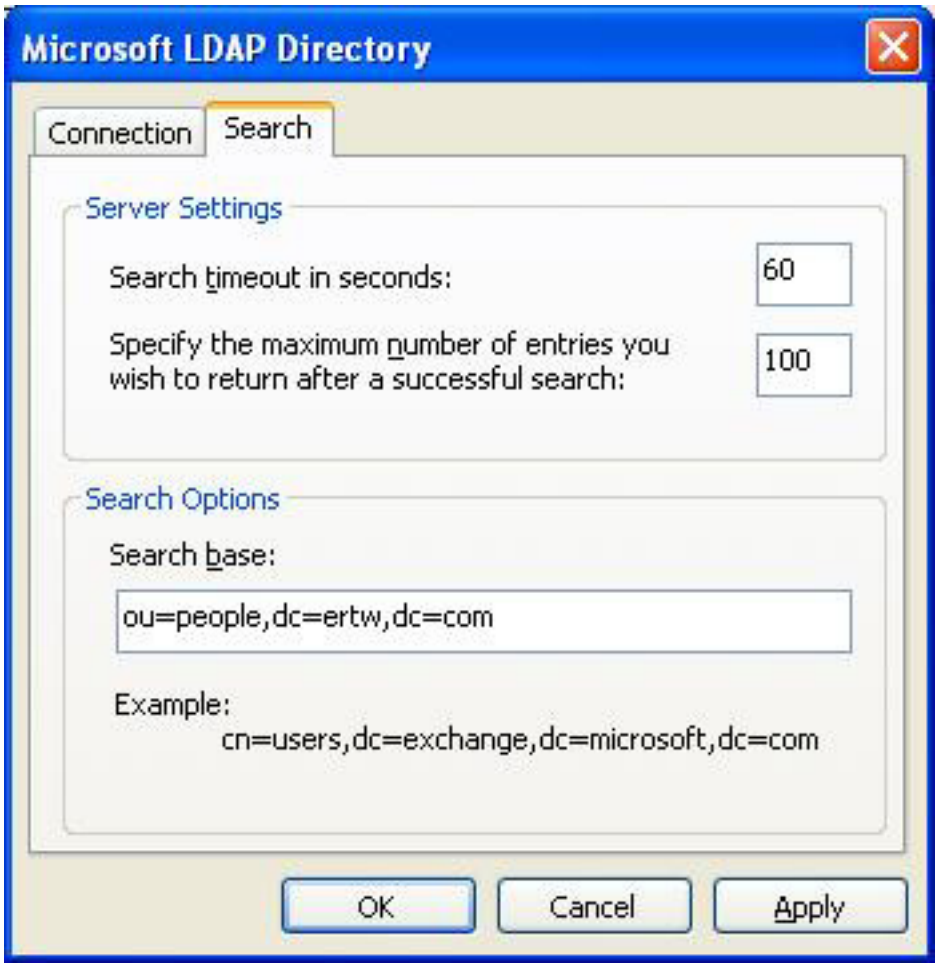


Figure 5 shows more options, the important one being the search base. Click **OK** after entering the search base, and you will be returned to the main Outlook screen.

You may now use the LDAP database wherever you are prompted to look for users by selecting the server name from the "Show Names From" field.

Section 5. Summary

In this tutorial you learned how to use your directory through searching and the command-line tools. You also learned how to configure e-mail clients to use the directory to store contact information.

Searching the LDAP tree requires you to build a query filter. The various operators that are used in a query are outlined in Table 3.

Table 3. LDAP Search Operators

Operator	Description	Example
=	Tests for equality	(cn=Walberg)
*	Tests for the existence of an attribute	(cn=*)
	Substring search	(sn=Walb*)
&	Logical AND	(&(condition1)(condition2))
|	Logical OR	(|(condition1)(condition2))
!	Logical NOT	(!(mail=*))
~=	"Sounds-like" match	(cn~=Shawn)
<= and >=	Range match	(pagesPerMinute >= 20)

Several utilities are provided to use the directory, such as `ldapsearch` for searching, and `ldapadd` and `ldapmodify` for adding and changing data. The tools that start with `ldap` operate through the LDAP protocol and require credentials to log in to the server. The tools that start with `slap` are used by the administrator and operate directly on the database.

This tutorial and the previous ones in the 301 series have focused on managing and working with an LDAP server. The next tutorial in the series will look at various applications including e-mail servers, and show how to use LDAP as the data source.

Resources

Learn

- Review the previous tutorial in this 301 series, "LPI exam 301 prep, Topic 303: Configuration" (developerWorks, March 2008), or all tutorials in the 301 series.
- Review the entire LPI exam prep tutorial series on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the LPIC Program, find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In RFC 4515 - The String Representation of LDAP Search Filters, find more info on search filters.
- Read "MS Outlook: What LDAP Attributes Are Recognised?" and "MS Outlook: How Do LDAP Attributes Map to Address Book Fields?" to see the results of reverse-engineering the attributes that are used by the Microsoft Outlook client.
- "LDAP for Rocket Scientists" is excellent, despite being a work in progress.
- Learn more about matching rules.
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux tips and Linux tutorials on developerWorks.
- Stay current with developerWorks technical events and Webcasts.

Get products and technologies

- The Firewall Builder utility makes the task of typing in iptables rules easy; it has a nice GUI and suite of tools to roll out updates to your firewalls.
- OpenLDAP is a great choice if you're looking for an LDAP server.
- phpLDAPadmin is a Web based LDAP administration tool. If a GUI is more your style, Luma is a good one to look at.
- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the developerWorks community through blogs, forums,

podcasts, and community topics in our new developerWorks spaces.

About the author

Sean Walberg
Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.